



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE

United States Patent and Trademark Office

Address: COMMISSIONER FOR PATENTS

P.O. Box 1450

Alexandria, Virginia 22313-1450

www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/798,474	03/10/2004	Mark Vincent Scardina	50277-2389	7416
43425 7590 11/13/2008 HICKMAN PALERMO TRUONG & BECKER/ORACLE 2055 GATEWAY PLACE SUITE 550 SAN JOSE, CA 95110-1083				
EXAMINER				
TRAN, QUOC A				
ART UNIT		PAPER NUMBER		
2176				
MAIL DATE		DELIVERY MODE		
11/13/2008		PAPER		

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary

Application No.

10/798,474

Applicant(s)

SCARDINA ET AL.

Examiner

Quoc A. Tran

Art Unit

2176

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --
Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 21 August 2008.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-6, 13-20, 22-25, 39-44, 48-61 and 63-69 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

- 5) ☐ Claim(s) _____ is/are allowed.

- 6) ☒ Claim(s) 1-6, 13-20, 22-25, 39-44, 48-61 and 63-69 is/are rejected.

- 7) ☐ Claim(s) _____ is/are objected to.

- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 21 August 2008 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
a) ☐ All b) ☐ Some * c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
2. ☐ Certified copies of the priority documents have been received in Application No. _____.
3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- 1) ☒ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftperson's Patent Drawing Review (PTO-948)
- 3) ☐ Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date _____
- 4) ☐ Interview Summary (PTO-413)
Paper No(s)/Mail Date _____
- 5) ☐ Notice of Informal Patent Application
- 6) ☐ Other: _____

Applicant's Response

This action is a Non-Final rejection in response to Applicant's RCE/Amendment/Remarks filed on 08/21/2008.

Claims 1-6, 13-20, 22-25, 39, 42-44, 48-61, 63-66, and 69 have been amended. Claims 7-12, 21, 26-38, 45-47 and 62 have been canceled. Claims 1-6, 13-20, 22-25, 39-44, 48-61, and 63-69 are pending in the present application. Effective filing date is 03/10/2004, priority date **09/04/2003** (Assignee Oracle).

Continued Examination Under 37 CFR 1.114

A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 08/21/2008 has been entered.

Claim Rejections - 35 USC § 101

35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claims 39-41, 48-61 and 63-69 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

Claims 39-41:

Claims 39-41 recite a "**volatile or non-volatile medium**" storing instruction for validating the streaming XML data -see the claims Pages 7-8. The Examiner notes the disclosure of the present invention expressly states, "*The term **computer-readable medium** as used herein refers to any medium that participates in providing instructions to processor 504 for execution. Such a medium may take many forms, including but not limited to, non-volatile media, **volatile media, and transmission media. Non-volatile media** includes, for example, optical or magnetic disks, such as storage device 510. **Volatile media** includes dynamic memory, such as main memory 506. **Transmission media** includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 502. **Transmission media** can also take the form of **acoustic or light waves**, such as those generated during **radio-wave** and **infra-red data communications.**"* see Specification → Page 16, Para [0057]. As such, the Claims 39-41 are drawn to a form of energy. Energy is not one of the four categories of invention and therefore this claim(s) is/are not statutory. Energy is not a series of steps or acts and thus is not a process. Energy is not a physical article or object and as such is not a machine or manufacture. Energy is not a combination of substances and

therefor not a composition of matter. Accordingly, Claims 39-41 fail to recite statutory subject matter, as defined in 35 U.S.C. 101.

Claims 48-61 and 63-69:

Claims 48-61 and 63-69 recite a "computer-readable storage medium" storing instruction for validating the streaming XML data -see the claims Pages 9-16. The Examiner notes the disclosure of the present invention expressly states, " *The term "computer-readable medium" as used herein refers to any medium that participates in providing instructions to processor 504 for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device 510. Volatile media includes dynamic memory, such as main memory 506. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 502. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications.*" see Specification → Page 16, Para [0057]. As such, the Claims 48-61 and 63-69 are drawn to a form of energy. Energy is not one of the four categories of invention and therefore this claim(s) is/are not statutory. Energy is not a series of steps or acts and thus is not a process. Energy is not a physical article or object and as such is not a machine or manufacture. Energy is not a combination of substances and therefor not a composition of matter. Accordingly, Claims 39-41 fail to recite statutory subject matter, as defined in 35 U.S.C. 101.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

Claims 1-6, 13-20, 22-25, 39-44, 48-61, and 63-69, are rejected under 35

U.S.C. 103(a) as being unpatentable over **Fry** US 20030159112A1, filed 11/26/2002 (hereinafter Fry, in view of in view of **Chu et al.** US 20050039124A1 filed 05/31/2001 (hereinafter Sijacic).

Independent claim 1,

Fry teaches:

A method comprising the computer-implemented steps of: while an XML processor performs a validation operation on an XML-based input stream, wherein said XML processor is configured to send validated XML data to an application,

(See the Abstract and Para [0022]→[0027]→ Fry disclose an XML document, an XML parser, XML processor, or XML reader in order to gain access to the document wherein the XML parser can then provide a variety of types of access to the application or client that does not require the entire document to be read into memory, including providing an XML stream, pulling XML information, and skipping unwanted XML from the

document [e.g. a single pass validation]. Also Fry further disclose at Para [0011], streaming API for XML parsing can be implemented on top of SAX, and constructs an easily manipulated event stream that is available to the application programmer. This allows the programmer to ask for the next event, or pull the event, rather than handling the event in a callback [e.g. while an XML processor performs a ...configured to send validated XML data to an application].)

In addition, Fry does not explicitly teach, but Chu teaches:

performing the steps of: while validating a particular XML element in said XML-based input stream, performing the computer implemented steps of: said XML processor receiving a request for particular information relating to said validation operation, wherein said request includes

AT LEAST ONE OF:

- (a) a request for whether said particular XML element is defined in corresponding information that dictates the structure of said XML data in said XML-based input stream;*
- (b) a request for the name of said particular XML element;*
- (c) a request for the data type of said particular XML element;*
- (d) a request for whether said particular XML element conforms to the corresponding information that dictates the structure of said XML data in said XML-based input stream;*

(e) a request for the current validation mode of said validation operation;

(f) a request for the current state of said validation operation; or

(g) a request for one or more annotations that are associated with said particular XML element;

(See at the Abstract and at Para [0027]→[0030] →Chu discloses this limitation that is the XML validating parser is validating the source document for detection of syntax errors, **at the same time** [e.g. while validating], **casting objects** [(c) a request for the data type of said particular XML element] and events from the parsed document to a level requested by the consumer application allows the consumer to receive only those objects or events for which it is adapted, without requiring the consumer application to include extra code to deal with objects or events it does not recognize in the parser's output [the parser simply discards those objects or events which this consumer is not interested in receiving].)

said XML processor generating one or more messages that include said particular information indicate to the said XML processor responding to said request for said particular information by providing said one or more messages.

(See Para [0053] →Chu discloses this limitation that is the parser instance where the parse method is invoked on the parser instance. The overridden parse method recognizes that the feature has been set, and retrieves the name that is specified for the desired abstraction level and passes that name to the superclass upon invocation. The

superclass then uses that abstraction level; typically identify the input document and where to print any error messages.)

Accordingly, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to have modified Fry's XML SAX Streaming parser API to include a means of said causing said XML processor to generate one or more messages that indicate to the application a request for the data type of said particular XML element as taught by Chu, because Fry and Chu are analogous art, since they are from the same field of endeavor of XML parsing and validating input xml data stream, and provides a predictable result of allows the a SAX or DOM parser includes name of a selected element to be passed to the method. The base parser can begin processing the XML document to locate an element tag signifying an element of the XML document. The iterative method can then direct the base parser to step through the elements in the document until the tag is located that corresponds to the selected element. The base parser can extract the selected element from the XML document and process the element such as by generating an event that can be read by a Java application. The event can then be placed on an event stream for use by an application- See Fry at Para 17.)

*Independent **claim 39,***

Claim 39 recites a computer -readable medium store instruction to implement a method recited in Claim 1. Thus, Fry and Chu disclose every

limitation of Claim 39 and provide proper reasons to combine, as indicated in the above rejections for Claim 1, see also Fry at Para 10, discloses memory (i.e. computer-readable medium.)

*Independent **Claim 48:***

Claim 48 recites a computer-readable storage medium store instruction to implement a method recited in Claim 1. Thus, Fry and Chu disclose every limitation of Claim 48 and provide proper reasons to combine, as indicated in the above rejections for Claim 1, see also Fry at Para 10, discloses memory (i.e. computer-readable medium.)

Claim 2,

Fry and Chu teach the method of claim 1 and further comprise:

wherein the step of causing said XML processor to generate one or more messages is performed in response to said request.

(See Para [0053] → Chu discloses this limitation that is the parser instance where the parse method is invoked on the parser instance. The overridden parse method recognizes that the feature has been set, and retrieves the name that is specified for the desired abstraction level and passes that name to the superclass upon invocation. The superclass then uses that abstraction level; typically identify the input document and where to print any error messages.)

Accordingly, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to have modified Fry's XML SAX Streaming parser API to include a means of said causing said XML processor to generate one or more messages is performed in response to said request as taught by Chu, because Fry and Chu are analogous art, since they are from the same field of endeavor of XML parsing and validating input xml data stream, and provides a predictable result of allows the a SAX or DOM parser includes name of a selected element to be passed to the method. The base parser can begin processing the XML document to locate an element tag signifying an element of the XML document. The iterative method can then direct the base parser to step through the elements in the document until the tag is located that corresponds to the selected element. The base parser can extract the selected element from the XML document and process the element such as by generating an event that can be read by a Java application. The event can then be placed on an event stream for use by an application- See Fry at Para 17.)

Claim 3,

Fry and Chu teach the method of claim 1 and further comprise:

wherein the step of said XML Processor generating said request for said particular information includes receiving said request via an application program interface through which information about said validation operation can be requested by an the application.

(See Para 8 --> Fry described the eXtensible Markup Language (XML) has become a standard for inter-application communication wherein XML messages passing between applications contain tags with self-describing text. The self-describing text allows these messages to be understandable not only to the applications, but also to humans reading an XML document.

Also see Fig. 1 and Para 22-27, 32 and 37→ Fry further the XML processing, forming the base class for all XML processors in the parsing paradigm, including for example the StreamParser and SAXDriver (generating SAX events and implement an XMLReader class from SAX see Para 37). The base parser iterates over XML Elements, which can then be encapsulated in the Element class that enforcing higher-level well-formedness constraints, such as proper element nesting and proper namespace declaration)

Claim 4,

Fry and Chu teach the method of claim 1 and further comprise:

wherein the step of said XML processor to generate one or more messages includes causing said XML processor generating said one or more messages that are transmitted in an output stream.

(See Para [0053] →Chu discloses this limitation that is the superclass then uses that abstraction level; typically identify the input document and where to print any error messages. Also Chu further discloses selecting an abstraction level to use when generating parser output by requesting generation of parser output, by a parser that

parses an input, such that the generated output adheres to a different syntax level than a syntax level used when validating the input - see Chu at Para [0026].)

Accordingly, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to have modified Fry's XML SAX Streaming parser API to include the step of said XML processor to generate one or more messages includes causing said XML processor generating said one or more messages that are transmitted in an output stream as taught by Chu, because Fry and Chu are analogous art, since they are from the same field of endeavor of XML parsing and validating input xml data stream, and provides a predictable result of allows the a SAX or DOM parser includes name of a selected element to be passed to the method. The base parser can begin processing the XML document to locate an element tag signifying an element of the XML document. The iterative method can then direct the base parser to step through the elements in the document until the tag is located that corresponds to the selected element. The base parser can extract the selected element from the XML document and process the element such as by generating an event that can be read by a Java application. The event can then be placed on an event stream for use by an application- See Fry at Para 17.)

Claim 5,

Fry and Chu teach the method of claim 1 and further comprise:

wherein the step of said XML processor generating one or more messages includes causing said XML processor to generate said one or more messages before completion of said validation operation on said XML-based input stream,

(See Para [0053] → Chu discloses this limitation that is the superclass then uses that abstraction level; typically identify the input document and where to print any error messages. Also Chu further discloses selecting an abstraction level to use when generating parser output by requesting generation of parser output, by a parser that parses an input, such that the generated output adheres to a different syntax level than a syntax level used when validating the input - see Chu at Para [0026].)

Accordingly, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to have modified Fry's XML SAX Streaming parser API to include the step of said XML processor generating one or more messages includes causing said XML processor to generate said one or more messages before completion of said validation operation on said XML-based input stream as taught by Chu, because Fry and Chu are analogous art, since they are from the same field of endeavor of XML parsing and validating input xml data stream, and provides a predictable result of allows the a SAX or DOM parser includes name of a selected element to be passed to the method. The base parser can begin processing the XML document to locate an element

tag signifying an element of the XML document. The iterative method can then direct the base parser to step through the elements in the document until the tag is located that corresponds to the selected element. The base parser can extract the selected element from the XML document and process the element such as by generating an event that can be read by a Java application. The event can then be placed on an event stream for use by an application- See Fry at Para 17.)

Claim 6,

Fry and Chu teach the method of claim 1 and further comprise:

wherein said validation operation includes performing a validation operation on said particular XML element of said XML-based input stream;

(See Para 22-27→ Fry disclose SAX as a streaming parser.)

In addition, Fry does not explicitly teach, but Chu teaches:

wherein the step of said XML processor generating said one or more messages includes causing said XML processor to generate said one or more messages that indicate how to process said particular XML element, only if said particular XML element is determined valid based on said validation operation on said particular XML element,

(See Para [0053] →Chu discloses this limitation that is the superclass then uses that abstraction level; typically identify the input document and where to print any error messages. Also Chu further discloses selecting an abstraction level to use when

generating parser output by requesting generation of parser output, by a parser that parses an input, such that the generated output adheres to a different syntax level than a syntax level used when validating the input - see Chu at Para [0026].)

Accordingly, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to have modified Fry's XML SAX Streaming parser API to include the step of said XML processor generating said one or more messages includes causing said XML processor to generate said one or more messages that indicate how to process said particular XML element, only if said particular XML element is determined valid based on said validation operation on said particular XML element as taught by Chu, because Fry and Chu are analogous art, since they are from the same field of endeavor of XML parsing and validating input xml data stream, and provides a predictable result of allows the a SAX or DOM parser includes name of a selected element to be passed to the method. The base parser can begin processing the XML document to locate an element tag signifying an element of the XML document. The iterative method can then direct the base parser to step through the elements in the document until the tag is located that corresponds to the selected element. The base parser can extract the selected element from the XML document and process the element such as by generating an event that can be read by a Java application. The event can then be placed on an event stream for use by an application- See Fry at Para 17.)

Claim 13,

Fry and Chu teach the method of claim 1 and further comprise:

*wherein said particular information, which is included in said one or more messages, comprises **ONE OR MORE OF:***

first data indicating whether said particular XML element is defined in the corresponding information that dictates the structure of said XML data in said XML-based input stream;

the name of the particular XML element that is currently being processed; the data type of the particular XML element that is currently being processed;

second data indicating whether said particular XML element conforms to the corresponding information that dictates the structure of said XML data in said XML-based input stream;

the current validation mode for the node particular XML element that is currently being processed, wherein the current validation mode is one of strict mode, lax mode, and skip mode; the current state of said validation operation;

or the one or more annotations that are associated with the particular XML element that is currently being processed.

(See Chu at Para [0046]--> discloses an event-based parser [SAX or DOM] is used, which contains nodes or objects only for the selected abstraction level; thereby perform type casting of objects on a selectable, dynamically-variable level.

Also see Chu at Para [0053] →discloses is the superclass then uses that abstraction level; typically identify the input document and where to print any error messages. Also Chu further discloses selecting an abstraction level to use when generating parser output by requesting generation of parser output, by a parser that parses an input, such that the generated output adheres to a different syntax level than a syntax level used when validating the input - see Chu at Para [0026].)

Accordingly, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to have modified Fry's XML SAX Streaming parser API to include the step of said the name of the particular XML element that is currently being processed; the data type of the particular XML element that is currently being processed as taught by Chu, because Fry and Chu are analogous art, since they are from the same field of endeavor of XML parsing and validating input xml data stream, and provides a predictable result of allows the a SAX or DOM parser includes name of a selected element to be passed to the method. The base parser can begin processing the XML document to locate an element tag signifying an element of the XML document. The iterative method can then direct the base parser to step through the elements in the document until the tag is located that corresponds to the selected element. The base parser can extract the selected element from the XML document and process the element such as by generating an event that can be read by a Java application. The event can then be placed on an event stream for use by an application- See Fry at Para 17.)

Claim 14,

Fry and Chu teach the method of claim 1 and further comprise:

wherein the step of receiving a request includes receiving a request regarding whether a first element of said XML- based input stream is defined in corresponding information that dictates the structure of XML data.

(See Fig. 1 and Para 22-27, 32 and 37→ Fry discloses the XML processing, forming the base class for all XML processors in the parsing paradigm, including for example the StreamParser and SAXDriver (generating SAX events and implement an XMLReader class from SAX see Para 37). The base parser iterates over XML Elements, which can then be encapsulated in the Element class that enforcing higher-level well-formedness constraints, such as proper element nesting and proper namespace declaration [e.g. dictates the XML structure data].)

Claim 15,

Fry and Chu teach the method of claim 1 and further comprise:

wherein the step of said XML processor receiving said request for said particular information includes receiving a request regarding what data type definition is associated with said particular XML element of said XML-based input stream, wherein said data type is defined in information that dictates the structure of corresponding XML data

(See Fig. 1 and Para 22-27, 32 and 37→ Fry discloses the XML processing, forming the base class for all XML processors in the parsing paradigm, including for example the StreamParser and SAXDriver (generating SAX events and implement an XMLReader class from SAX see Para 37). The base parser iterates over XML Elements, which can then be encapsulated in the Element class that enforcing higher-level well-formedness constraints, such as proper element nesting and proper namespace declaration [e.g. dictates the XML structure data].

Also see Fry at Para [0020], described validating with XML schemas or Document Type Definitions (DTDs). These parsers can be selected by instantiating different implementations of a streaming XML API. A pull-parser streaming API can also support data binding implementations.)

Claim 16,

Fry and Chu teach the method of claim 1 and further comprise:

wherein the step of said XML processor receiving said request for said particular information includes receiving a request regarding what data type definition is associated with an attribute of said particular XML element, wherein said data type that is associated with said attribute is defined in said information that dictates the structure of corresponding XML data.

(See Fig. 1 and Para 22-27, 32 and 37→ Fry discloses the XML processing, forming the base class for all XML processors in the parsing paradigm, including for example

the StreamParser and SAXDriver (generating SAX events and implement an XMLReader class from SAX see Para 37). The base parser iterates over XML Elements, which can then be encapsulated in the Element class that enforcing higher-level well-formedness constraints, such as proper element nesting and proper namespace declaration [e.g. dictates the XML structure data].

Also see Fry at Para [0020], described validating with XML schemas or Document Type Definitions (DTDs). These parsers can be selected by instantiating different implementations of a streaming XML API. A pull-parser streaming API can also support data binding implementations.)

Claim 17,

Fry and Chu teach the method of claim 1 and further comprise:

wherein the step of said XML processor receiving said request for said particular information includes receiving a request regarding whether a data type of content of said particular XML element of said XML-based input stream conforms to a corresponding data type definition in information that dictates the structure of corresponding XML data.

(See Fig. 1 and Para 22-27, 32 and 37→ Fry discloses the XML processing, forming the base class for all XML processors in the parsing paradigm, including for example the StreamParser and SAXDriver (generating SAX events and implement an XMLReader class from SAX see Para 37). The base parser iterates over XML Elements, which can then be encapsulated in the Element class that enforcing higher-

level well-formedness constraints, such as proper element nesting and proper namespace declaration [e.g. dictates the XML structure data].

Also see Fry at Para [0024]→[0030], discloses element type "two" in the XML document, corresponding to another StartElementEvent: element 310 in the event stream. This would generate a substream in the Java environment to handle the second element type. Values 312, 314, 316 of element type "two" are placed onto the event stream and correspond to the Java substream. Element type "two" ends when another end tag is reached in the document, corresponding to an EndElementEvent 318 in the event stream, with another EndElementEvent 320 corresponding to the end of document tag </doc>..)

Claim 18,

Fry and Chu teach the method of claim 1 and further comprise:

wherein the step of said XML processor receiving said request for said particular information includes receiving a request regarding a first annotation that is associated with said particular XML element of said XML-based input stream, wherein said first annotation is defined in information that dictates the structure of corresponding XML data.

(See Fig. 1 and Para 22-27, 32 and 37→ Fry discloses the XML processing, forming the base class for all XML processors in the parsing paradigm, including for example the StreamParser and SAXDriver (generating SAX events and implement an XMLReader class from SAX see Para 37). The base parser iterates over XML

Elements, which can then be encapsulated in the Element class that enforcing higher-level well-formedness constraints, such as proper element nesting and proper namespace declaration [e.g. dictates the XML structure data].

Also see Fry at Para [0043] --> [0044], discloses XML-based input stream, wherein said a particular abstraction level, or "type casting", is defined in information that dictates the structure of corresponding XML data validating with XML [e.g. annotations associated with elements].)

Claim 19,

Fry and Chu teach the method of claim 18 and further comprise:

*wherein said information that dictates the structure of
corresponding XML data comprises a second annotation definition that is
associated with a second XML element of said XML-based input stream
that is different than said particular XML element, and wherein the step of
said XML processor receiving said request for said particular information
includes receiving a request regarding said second annotation,*

(See Fig. 1 and Para 22-27, 32 and 37→ Fry discloses the XML processing, forming the base class for all XML processors in the parsing paradigm, including for example the StreamParser and SAXDriver (generating SAX events and implement an XMLReader class from SAX see Para 37). The base parser iterates over XML Elements, which can then be encapsulated in the Element class that enforcing higher-

level well-formedness constraints, such as proper element nesting and proper namespace declaration [e.g. dictates the XML structure data].

Also see Fry at Para [0043] --> [0044], discloses XML-based input stream, wherein said a particular abstraction level, or "type casting", is defined in information that dictates the structure of corresponding XML data validating with XML [e.g. annotations associated with elements].)

*the method further comprising the computer- implemented step of:
before responding to said request regarding said second annotation,
responding to a request regarding whether said particular XML element is
defined in said information that dictates the structure of corresponding
XML data.*

(See Fig. 1 and Para 22-27, 32 and 37→ Fry discloses the XML processing, forming the base class for all XML processors in the parsing paradigm, including for example the StreamParser and SAXDriver (generating SAX events and implement an XMLReader class from SAX see Para 37). The base parser iterates over XML Elements, which can then be encapsulated in the Element class that enforcing higher-level well-formedness constraints, such as proper element nesting and proper namespace declaration [e.g. dictates the XML structure data].

Also see Fry at Para [0043] --> [0044], discloses XML-based input stream, wherein said a particular abstraction level, or "type casting", is defined in information

that dictates the structure of corresponding XML data validating with XML [e.g. annotations associated with elements].)

Claim 20,

Fry and Chu teach the method of claim 1 and further comprise:

wherein the step of said XML processor receiving said request for said particular information includes receiving a request regarding a status of said validation operation with respect to said particular XML element of said XML-based input stream.

(See Fry at Para [0009], described SAX, one writes handlers, or objects that implement the various handler APIs, which receive callbacks during the processing of an XML document, wherein the SAX API the programmer must keep track of the current state of the document in the code each time one processes an XML document.)

Claim 22,

Fry and Chu teach the method of claim 1 and further comprise:

wherein the step of said XML processor receiving said request for said particular information includes receiving a request from an event handler sent in response to an event received in a parser output stream.

(See Para [0053] → Chu discloses this limitation that is the superclass then uses that abstraction level; typically identify the input document and where to print any error messages. Also Chu further discloses selecting an abstraction level to use when

generating parser output by requesting generation of parser output, by a parser that parses an input, such that the generated output adheres to a different syntax level than a syntax level used when validating the input - see Chu at Para [0026].)

Accordingly, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to have modified Fry's XML SAX Streaming parser API to include the step of said XML processor receiving said request for said particular information includes receiving a request from an event handler sent in response to an event received in a parser output stream as taught by Chu, because Fry and Chu are analogous art, since they are from the same field of endeavor of XML parsing and validating input xml data stream, and provides a predictable result of allows the a SAX or DOM parser includes name of a selected element to be passed to the method. The base parser can begin processing the XML document to locate an element tag signifying an element of the XML document. The iterative method can then direct the base parser to step through the elements in the document until the tag is located that corresponds to the selected element. The base parser can extract the selected element from the XML document and process the element such as by generating an event that can be read by a Java application. The event can then be placed on an event stream for use by an application- See Fry at Para 17.)

Claim 23,

Fry and Chu teach the method of claim 1 and further comprise:

wherein the step of said XML processor responding to said request includes providing, in an output stream, said particular information.

(See Para [0053] → Chu discloses this limitation that is the superclass then uses that abstraction level; typically identify the input document and where to print any error messages. Also Chu further discloses selecting an abstraction level to use when generating parser output by requesting generation of parser output, by a parser that parses an input, such that the generated output adheres to a different syntax level than a syntax level used when validating the input - see Chu at Para [0026].)

Accordingly, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to have modified Fry's XML SAX Streaming parser API to include wherein the step of said XML processor responding to said request includes providing, in an output stream, said particular information as taught by Chu, because Fry and Chu are analogous art, since they are from the same field of endeavor of XML parsing and validating input xml data stream, and provides a predictable result of allows the a SAX or DOM parser includes name of a selected element to be passed to the method. The base parser can begin processing the XML document to locate an element tag signifying an element of the XML document. The iterative method can then direct the base parser to step through the elements in the document until the tag is located that corresponds to the selected element. The base parser can extract the selected

element from the XML document and process the element such as by generating an event that can be read by a Java application. The event can then be placed on an event stream for use by an application- See Fry at Para 17.)

Claim 24,

Fry and Chu teach the method of claim 18 and further comprise:

further comprises the computer implemented step of: parsing said XML-based input stream only once for both of said validation operation and operations that are dictated by annotations associated with elements in said XML-based input stream.

(See Fig. 1 and Para 22-27, 32 and 37 → Fry discloses the XML processing, forming the base class for all XML processors in the parsing paradigm, including for example the StreamParser and SAXDriver (generating SAX events and implement an XMLReader class from SAX see Para 37). The base parser iterates over XML Elements, which can then be encapsulated in the Element class that enforcing higher-level well-formedness constraints, such as proper element nesting and proper namespace declaration [e.g. dictates the XML structure data].

Also see Fry at Para [0043] --> [0044], discloses XML-based input stream, wherein said a particular abstraction level, or "type casting", is defined in information that dictates the structure of corresponding XML data validating with XML [e.g. annotations associated with elements.]

Claim 25,

Fry and Chu teach the method of claim 18 and further comprise:

wherein information that dictates the structure of corresponding said XML data in said XML-based input stream, with which said input stream is validated in said validation operation, comprises a plurality of schema definitions that are associated with a plurality of corresponding XML documents that could be constituent to said XML-based input stream.

(See Fig. 1 and Para 22-27, 32 and 37→ Fry discloses the XML processing, forming the base class for all XML processors in the parsing paradigm, including for example the StreamParser and SAXDriver (generating SAX events and implement an XMLReader class from SAX see Para 37). The base parser iterates over XML Elements, which can then be encapsulated in the Element class that enforcing higher-level well-formedness constraints, such as proper element nesting and proper namespace declaration [e.g. dictates the XML structure data].

Also see Fry at Para [0020], described validating with XML schemas or Document Type Definitions (DTDs). These parsers can be selected by instantiating different implementations of a streaming XML API. A pull-parser streaming API can also support data binding implementations.)

Claim 40,

Claim 40 recite a computer-readable storage medium store instruction to implement a method recited in Claims 13-15. Thus, Fry and Sijacic disclose every limitation of Claim 40 and provide proper reasons to combine, as indicated in the above rejections for Claims 13-15, see also Fry at Para 10, discloses memory (i.e. computer-readable medium.)

Claim 41,

Claim 41 recites a computer-readable storage medium store instruction to implement a method recited in Claims 13 and 22. Thus, Fry and Sijacic disclose every limitation of Claim 41 and provide proper reasons to combine, as indicated in the above rejections for Claims 13 and 22, see also Fry at Para 10, discloses memory (i.e. computer-readable medium.)

Claims 42-43,

Fry and Chu teach the method of claim 1 and further comprise:

reading said one or more metadata, XML schema that corresponds to said XML-based input stream, reading said one or more annotation from metadata, XML schema that corresponds to said XML-based input stream that corresponds to said XML based input stream,

(See Para 22-27→ Fry disclose SAX as a streaming parser (i.e. XML validator in streaming fashion, wherein the data being validated while streaming.

Also see Fry at Para [0043] --> [0044], discloses XML-based input stream, wherein said a particular abstraction level, or "type casting", is defined in information that dictates the structure of corresponding XML data validating with XML [e.g. annotations associated with elements.]

Claim 44,

Fry and Chu teach the method of claim 1 and further comprise:

wherein the step of causing said XML processor to generate generating said one or more messages includes causing said XML processor to generate said one or more messages that indicate to the application how to conform said particular XML element to one or more requirements of the application that uses said particular XML element.

(See Chu at Para [0046]--> discloses an event-based parser [SAX or DOM] is used, which contains nodes or objects only for the selected abstraction level; thereby perform type casting of objects on a selectable, dynamically-variable level.

Also see Chu at Para [0053] →discloses is the superclass then uses that abstraction level; typically identify the input document and where to print any error messages. Also Chu further discloses selecting an abstraction level to use when generating parser output by requesting generation of parser output, by a parser that parses an input, such that the generated output adheres to a different syntax level than a syntax level used when validating the input - see Chu at Para [0026].)

Accordingly, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to have modified Fry's XML SAX Streaming parser API to include the step of causing said XML processor to generate generating said one or more messages includes causing said XML processor to generate said one or more messages that indicate to the application how to conform said particular XML element to one or more requirements of the application that uses said particular XML element as taught by Chu, because Fry and Chu are analogous art, since they are from the same field of endeavor of XML parsing and validating input xml data stream, and provides a predictable result of allows the a SAX or DOM parser includes name of a selected element to be passed to the method. The base parser can begin processing the XML document to locate an element tag signifying an element of the XML document. The iterative method can then direct the base parser to step through the elements in the document until the tag is located that corresponds to the selected element. The base parser can extract the selected element from the XML document and process the element such as by generating an event that can be read by a Java application. The event can then be placed on an event stream for use by an application- See Fry at Para 17.)

Claims 49-53 respectively:

Claims 49-53 recite a computer-readable storage medium store instruction to implement a method recited in Claims 2-6. Thus, Fry and Chu disclose every limitation of Claims 49-53 respectively and provide proper reasons to combine,

as indicated in the above rejections for Claims 2-6, see also Fry at Para 10, discloses memory (i.e. computer-readable medium.)

Claim 54:

Claim 54 recite a computer-readable storage medium store instruction to implement a method recited in Claim 13. Thus, Fry and Chu disclose every limitation of Claim 54 and provide proper reasons to combine, as indicated in the above rejections for Claim 13, see also Fry at Para 10, discloses memory (i.e. computer-readable medium.)

Claims 55-61 and 63-69 respectively:

Claim 55-61 and 63-69 recite a computer-readable storage medium store instruction to implement a method recited in Claims 14-20, 22-25 and 42-44 respectively. Thus, Fry and Chu disclose every limitation of Claims 55-61 and 63-69 respectively and provide proper reasons to combine, as indicated in the above rejections for Claims 14-20, 22-25 and 42-44, see also Fry at Para 10, discloses memory (i.e. computer-readable medium.)

It is noted that any citations to specific, pages, columns, lines, or figures in the prior art references and any interpretation of the references should not be considered to be limiting in any way. A reference is relevant for all it contains and may be relied upon

for all that it would have reasonably suggested to one having ordinary skill in the art.
See, MPEP 2123.

Response to Arguments

Applicant's arguments with respect to claims 1, 39, 48 and 2-6, 13-20, 22-25, 40-44, 49-61 and 63-69 have been considered but are moot in view of the new ground(s) of rejection (See the Remarks Page 17-23).

It is noted the Examiner introducing the Chu reference for the new ground of rejection presents in the current Office Action (See above rejection for details).

Furthermore, the Examiner retains Fry reference as discussed above, since Fry discloses the SAX as a streaming parser (i.e. XML validator in streaming fashion, wherein the data is being validated while streaming - See Fry at Para 22-27. In addition Sijacic discloses in fig. 5-6, which shown steps 610-660 (i.e. validating request, parsing, repores to message) wherein the request messages are parsed by event-based parser or API such as Simple API for XML (SAX) a parser operating within the XML DOM 510 (Step 620 Fig. 6), and further defines the logical structure of these documents and the manner by which they are edited and accessed. This structure, or model, enables XML servlet 222 to identify interfaces and objects used to represent and modify a document; the behavior and attributes of these interfaces and objects; and any relationships between the interfaces and object, See Sijacic in Fig. 5-6 and the Abstract and Para 11,

32, 43 and 58-62. Also Sijacic discloses the validating request, parsing, repeses to message wherein the request messages are parsed by event-based parser or API such as Simple API for XML (SAX) a parser operating within the XML DOM 510 (Step 620 Fig. 6) and Para 11, 32, 43 and 58-62, and further view of the above rejection for details.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Quoc A. Tran whose telephone number is 571-272-8664. The examiner can normally be reached on Mon through Fri 8AM - 5PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Doug Hutton can be reached on (571) 272-4137. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Quoc A. Tran/
Examiner, Art Unit 2176

/Doug Hutton/
Doug Hutton
Supervisory Primary Examiner
Technology Center 2100